



arc⁴² Reality Check

Praxiseinsatz, Werkzeuge,
Add-Ons

Dr. Gernot Starke
innoQ Fellow

WJAX. November 2015

 Follow @gernotstarke



Dr. Gernot Starke

innoQ Fellow



► **Softwarearchitekturen**

Entwurf, Entwicklung, Management
Evolution & Modernisierung
Dokumentation
Training

► **Mentoring und Coaching**

Analyse und Optimierung von
Entwicklungsprozessen

► **Reviews**

+49 177 – 728 2570
Gernot.Starke@innoQ.com

www.arc42.de



Was ist arc42?



Behälter für Architektur-Info

Anforderungen

Entscheidungen



Strukturen von
Sourcecode



Warum?





<http://www.flickr.com/photos/mistressf/737350025/sizes/l/>





Detail

Gallery

[Summary](#)
[Detail](#)

View

[By name](#)
[By source](#)

Browse

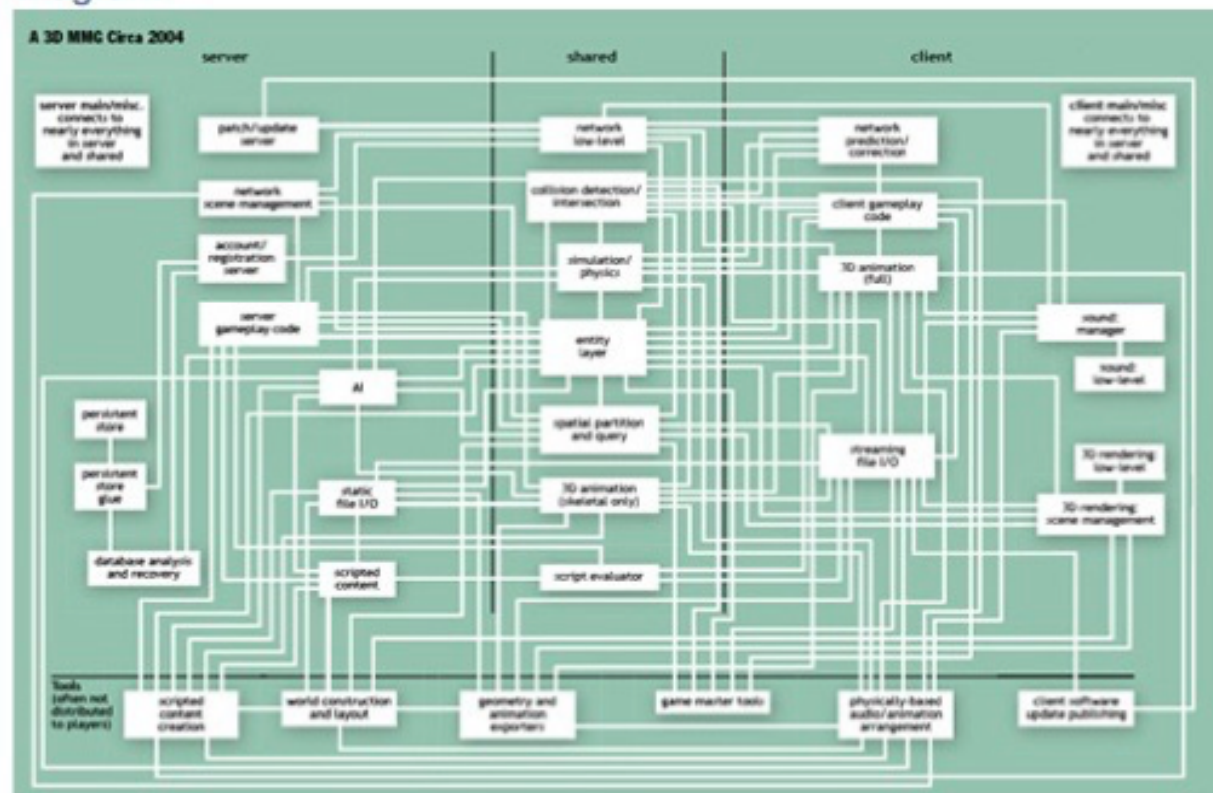
[By domain](#)
[By view](#)

Game 2004 MMORPG

Source

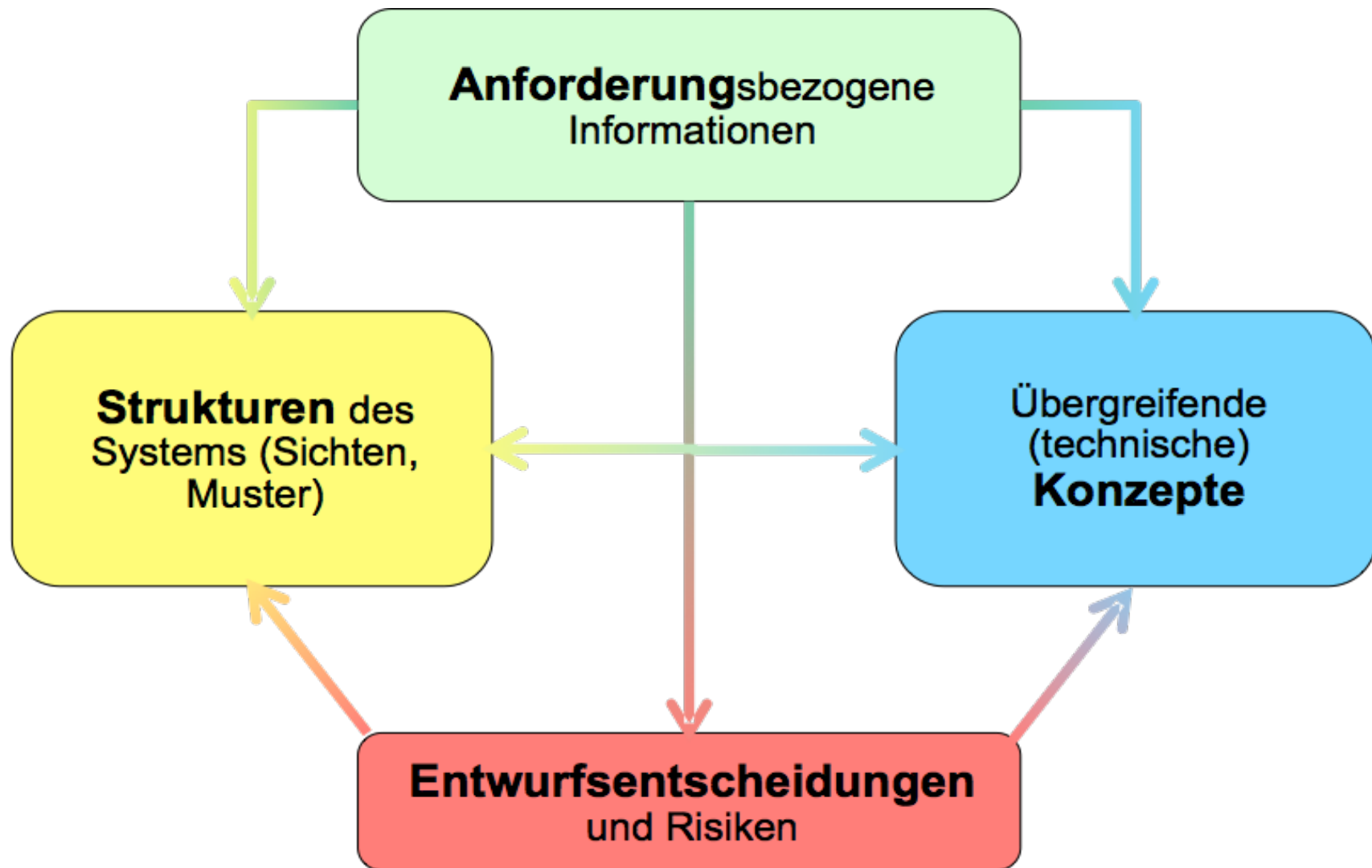
Blow, J. "Game Development: Harder Than You Think." *ACM Queue*, February 2004, p. 34.

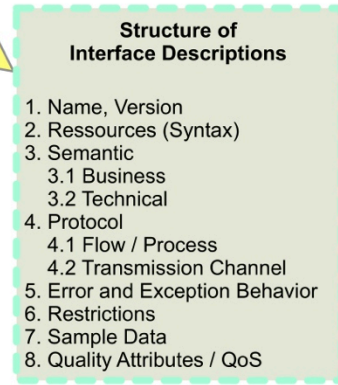
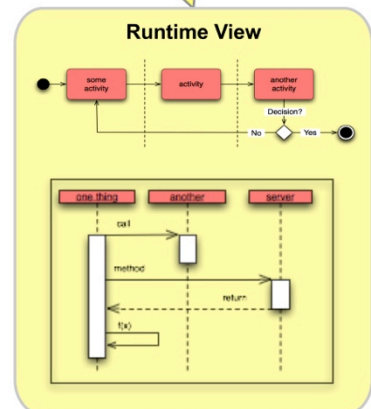
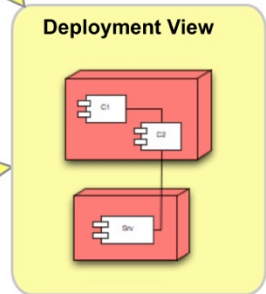
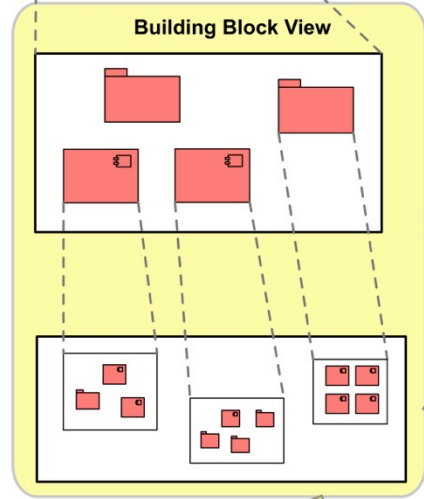
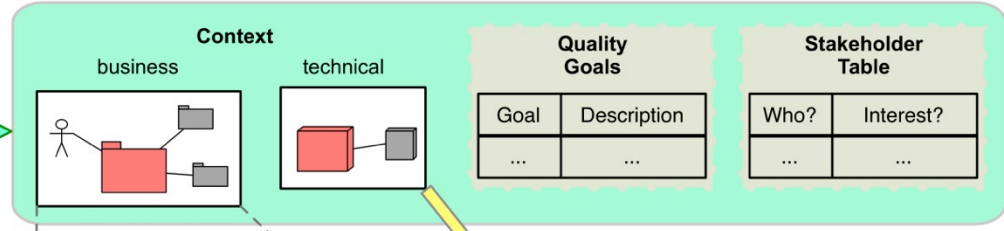
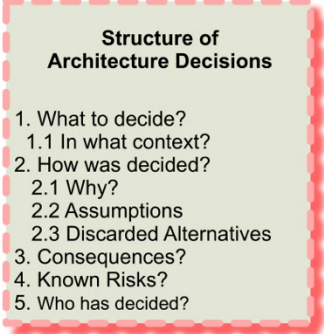
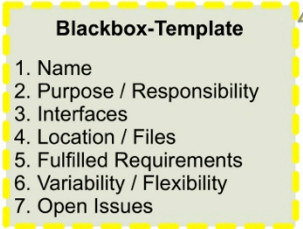
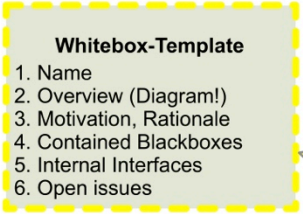
Diagram



Wie sieht's aus?







1. Einführung und Ziele

- 1.1 Aufgabenstellung
- 1.2 Qualitätsziele
- 1.3 Stakeholder

2. Randbedingungen

- 2.1 Technische Randbedingungen
- 2.2 Organisatorische Randbedingungen
- 2.3 Konventionen

3. Kontextabgrenzung

- 3.1 Fachlicher Kontext
- 3.2 Technischer- oder Verteilungskontext

4. Lösungsstrategie**5. Bausteinsicht**

- 5.1 Ebene 1
- 5.2 Ebene 2
-

6. Laufzeitsicht

- 6.1 Laufzeitszenario 1
- 6.2 Laufzeitszenario 2
-

7. Verteilungssicht

- 7.1 Infrastruktur Ebene 1
- 7.2 Infrastruktur Ebene 2
-

8. Konzepte

- 8.1 Fachliche Struktur und Modelle
- 8.2 Typische Muster und Strukturen
- 8.3 Persistenz
- 8.4 Benutzeroberfläche
-

9. Entwurfsentscheidungen

- 9.1 Entwurfsentscheidung 1
- 9.2 Entwurfsentscheidung 2
-

10. Qualitätsszenarien

- 10.1 Qualitätsbaum
- 10.2 Qualitäts-/Bewertungsszenarien

11. Risiken**12. Glossar**

2006...



delta

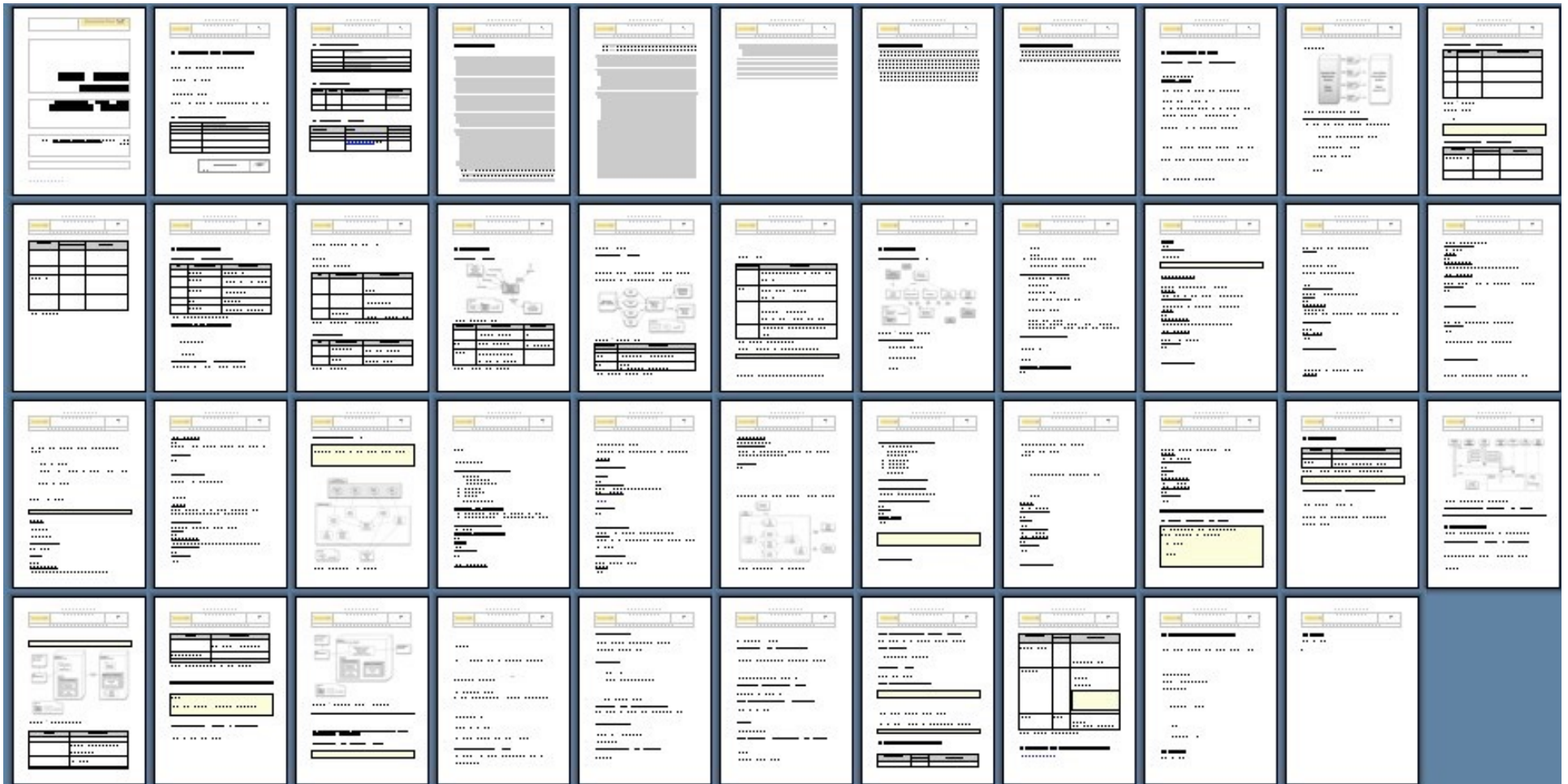
Beispiel (Datenmigration, 4000+ PT)

12

12.4 Beispiel einer Architektur dokumentation

The image displays a grid of 18 architectural diagrams and tables, organized into three rows and six columns. The top-left cell contains a large number '12' and the title '12.4 Beispiel einer Architektur dokumentation'. The diagrams include flowcharts, organizational charts, and data tables, illustrating the data migration process for 4000+ PT. The diagrams are arranged in a grid, with each cell containing a different diagram or table. The diagrams show various components and their relationships, including data flows, organizational structures, and data tables. The tables contain columns and rows of data, representing different aspects of the migration process. The flowcharts show the sequence of steps and the flow of information between different components. The organizational charts show the hierarchy and structure of the project. The diagrams are arranged in a grid, with each cell containing a different diagram or table. The diagrams show various components and their relationships, including data flows, organizational structures, and data tables. The tables contain columns and rows of data, representing different aspects of the migration process. The flowcharts show the sequence of steps and the flow of information between different components. The organizational charts show the hierarchy and structure of the project.

Beispiel (CRM-System, 2000+ PT)



Beispiel „htmlSanityCheck“

Prerequisites

To understand the project & SOURCECODE

- Basic knowledge of HTML (link, anchor, id, URI, ImageMap)
- Basic knowledge of Groovy + Gradle

Disclaimer:

- This project seriously overdocumented
- Code is unfinished !!

Intro and Business Goals

Goal (Output)

- HTML report
 - Summary
 - Results by page
 - Minimally styled

Intro (2): What can go wrong...

- Common HTML generators* guarantee only **syntactic correctness**
 - * Especially Markdown, AsciiDoc, AsciiDocer
- Potential semantic errors include:
 - Broken links, i.e.
 - Broken cross references
 - Broken ImageMaps
 - Missing images
 - Ambiguous definitions
 - Unused resources (i.e. image files)

Intro (3): Why can it go wrong?

- Markup languages simplify linking.
- Generated links sensitive to **spelling, typos, renames, deletions**

Requirements (2): Checks

ID	Check	Description
R1	Missing image file	Check all image tags if the referenced image file exists. See MissingImageFileChecker
R2	Broken external links	Check all external links whether they are "HTTP" or "file" links. See BrokenExternalLinkChecker
R3	Missing local file	Check all local links if an external file exists. See MissingLocalFileChecker
R4	ImageMap file missing	Check all ImageMap tags if the referenced image file exists. See ImageMapFileMissingChecker
R5	Additional data	Check all data attributes for additional content.
R6	Missing alt and title	Check if the alt and title attributes are set on referenced file or on the ImageMap. See MissingAltAndTitleChecker
R7	Unused images	Check if the image resources are not referenced by any of the ImageMap. See UnusedImageChecker

Requirements (1)

Read HTML file: Shall read HTML file with configurable name / location

Configurable to process multiple files: Shall be configurable for a set of files to be processed in a single run and produce a joint report. Useful for e.g. API documentation where each HTML file references each other.

Shall be usable as Groovy plugin: Shall be usable as Groovy plugin

Command line usage: Shall be usable from the command line with minimal prerequisites for installation.

Configurable output: Shall be configurable for the output format.

Available on public repository: Like, e.g. Maven, located on the Gradle Plugin Repository

Include suggestions: In case of e.g. broken cross references, shall provide suggestions what „should have been“ by applying using sanitycheck.

Architecture Quality Goals

Priority	Quality Goal	Scenario
1	Correctness	Events broken external link (cross reference) is found
1	Correctness	Events missing local image is found
1	Safety	Content of the files to be checked is never altered
1	Correctness	Content of every checked file is automatically tested for proper HTML compliance
1	Correctness	Events reporting format is robust. Reports must encode HTML checking results
1	Performance	Check of 200k HTMLs performed under 10 sec including gradle startup

Constraints

- Implement on Java platform
- Preferably in Groovy
- Runnable as Gradle-plugin without installation*
 - * except Java runtime
- Available under liberal Open-Source license

Context

Solution Strategy

- Implement in Groovy
 - Wrapping as Gradle-plugin becomes simple
- Apply Template-Method pattern for
 - Checking algorithms
 - Report generators
 (Details see chap 8: Crosscutting Concepts)

Building Blocks (Level 1)

Building Blocks (HSC Core, Level 2)

Building Blocks (Checker, Level 3)

Building Blocks (ResultsCollector, L3)

Building Blocks | Hierarchy

Gradle sub-projects:

- html-sanity-check
- html-sanity-check-gradle-plugin

gradle-plugins:

- html-sanity-check-gradle-plugin

Package:

- com.arc42.htmlsanitycheck
- com.arc42.htmlsanitycheck-gradle-plugin

Class:

- com.arc42.htmlsanitycheck.htmlsanitycheck
- com.arc42.htmlsanitycheck.htmlsanitycheck-gradle-plugin

Runtime (perform all checks)

Deployment

Crosscutting Concepts (Domain)

Crosscutting Concepts (Template Method)

The Template Method defines a skeleton of an algorithm in an operation, and defers some steps to subclasses.

```

http://sourceyarn.com/design_patterns/template_method

• Checking HTML [abstract]

public SingleCheckResult htmlPage pageToCheck ()
    @SuppressWarnings("null")
    checkingResults = new SingleCheckResult ()
    intoCheckingResultDescription ()
    return check ()
    
```

Decisions

- Jsoup as HTML parser (<http://jsoup.org>)
 - No [] external dependencies
 - Simple API, DOM-like navigation

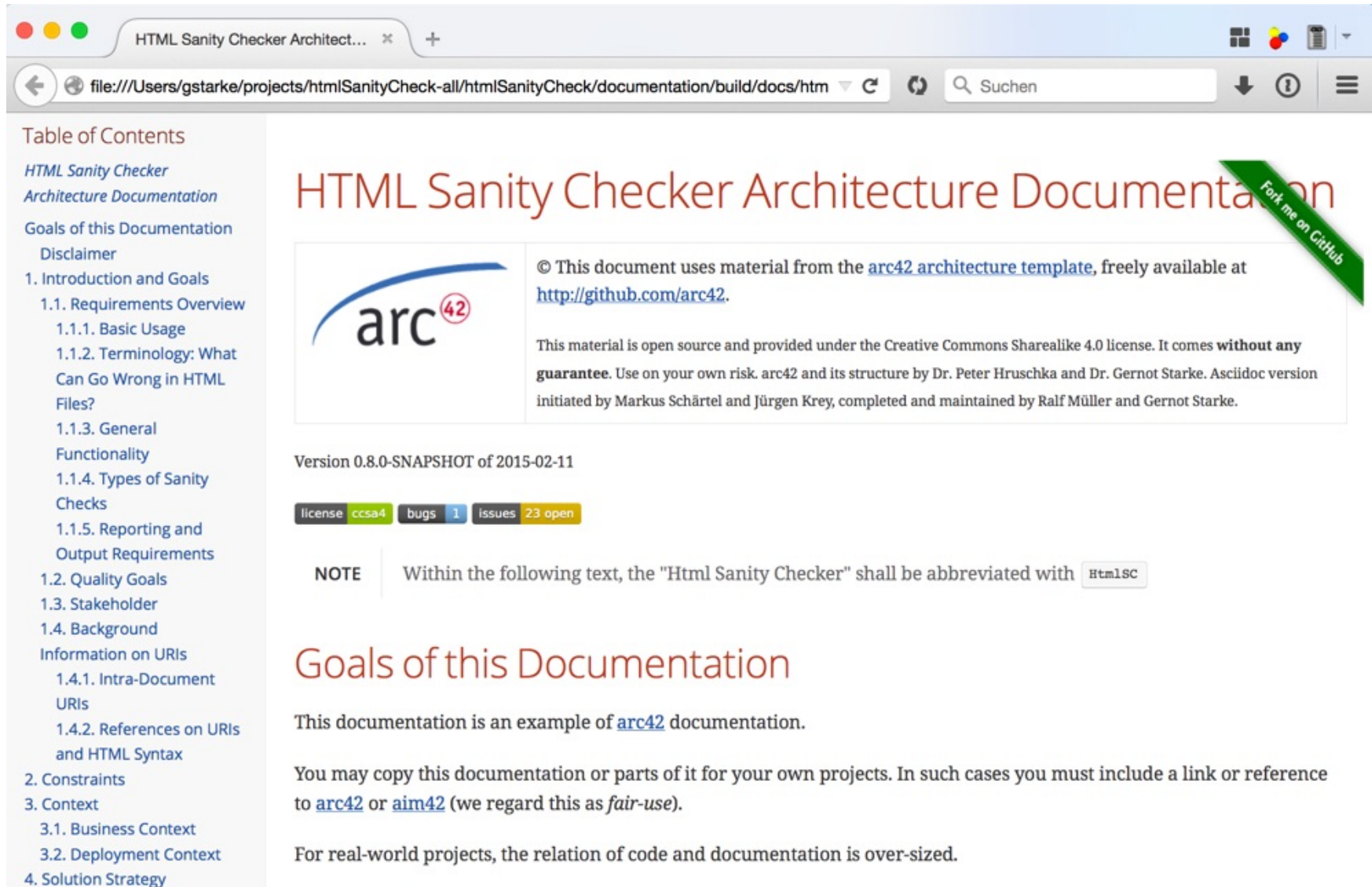
Open Issues

- Many...: <https://github.com/aim42/htmlsanitycheck/issues>

Glossary

- Anchor
- ID
- ImageMap
- Link
- Resource
- URI
- HtmlPage
- (Single) Check

Beispiel „htmlSanityCheck“ (2)



HTML Sanity Checker Architecture Documentation

© This document uses material from the [arc42 architecture template](http://github.com/arc42), freely available at <http://github.com/arc42>.

This material is open source and provided under the Creative Commons Sharealike 4.0 license. It comes **without any guarantee**. Use on your own risk. arc42 and its structure by Dr. Peter Hruschka and Dr. Gernot Starke. Asciidoc version initiated by Markus Schärtel and Jürgen Krey, completed and maintained by Ralf Müller and Gernot Starke.

Version 0.8.0-SNAPSHOT of 2015-02-11

license **ccsa4** bugs **1** issues **23 open**

NOTE Within the following text, the "Html Sanity Checker" shall be abbreviated with `HtmlSC`

Goals of this Documentation

This documentation is an example of [arc42](#) documentation.

You may copy this documentation or parts of it for your own projects. In such cases you must include a link or reference to [arc42](#) or [aim42](#) (we regard this as *fair-use*).

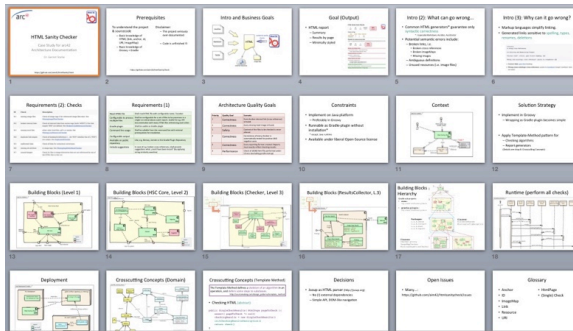
For real-world projects, the relation of code and documentation is over-sized.



Mit arc42 starten?



Mit arc42 starten (Tag 1)



3 Kontext

3.1 Fachlicher Kontext

Motivation
An den Schnittstellen zwischen einem System und seiner Umgebung treten Fehler besonders gerne und häufig auf. Die Schnittstellen zu Nachbarsystemen gehören zu den kritischsten Aspekten jedes Systems. Stellen Sie daher sicher, dass Sie die Außerachtlassungen komplett verstanden haben.

Was und wie
Der fachliche Kontext zeigt alle fachlichen Nachbarn des betrachteten Systems mit allen fachlichen Daten/Ereignissen/Nachrichten, die ausgetauscht werden. Zusätzlich annotiert werden eventuell Datenformate und Protokolle der Kommunikation mit Nachbarsystemen und der Umwelt (falls diese nicht erst bei den spezifischen Bausteinen präzisiert wird).

Tipps

- Besetzen Sie den fachlichen Kontext als Feedback-Instrument für möglichst viele der Stakeholder (siehe Kapitel 1.3).
- Stellen Sie durch dieses Feedback sicher, dass Sie alle (!) fachlichen Nachbarsysteme identifiziert haben.
- Suchen Sie an den externen Schnittstellen nach Risiken: Wo droht Ihrem System hohe Volatilität, hohe (inhaltliche oder technische) Komplexität, wo drohen besondere Kosten oder Aufwände?
- An Schnittstellen können besondere Qualitätsanforderungen gelten, etwa Sicherheits-, Durchsatz- oder Verfügbarkeitsanforderungen. Diese sogenannten „Service Levels“ können sehr hohe Implementierungs- und Betriebsaufwände und -risiken nach sich ziehen – daher ist besondere Vorsicht geboten.
- Markieren Sie erkannte Risiken im Kontextdiagramm (In unserem Beispiel (Abb. 3.1) kann die Prüfung externer Websites lange dauern und ist daher kritisch für unsere Performanceanforderung).

Abbildung 3.1: Fachlicher Kontext

Nachbar	Beschreibung
User	Erstellt Dokumentation mit einem Werkzeug, das HTML als Ausgabeformat liefert. Erhält vom System die Prüfergebnisse.
local html files	htmlSanityCheck liest und überprüft lokale HTML-Dokumente.
local images	htmlSanityCheck überprüft, ob referenzierte Bilder als Dateien (lokal) vorliegen bzw. ob Bilddateien überhaupt referenziert werden.
external websites & resources	htmlSanityCheck kann zusätzlich auch zur Prüfung externer Links verwendet werden.



Beispiele!

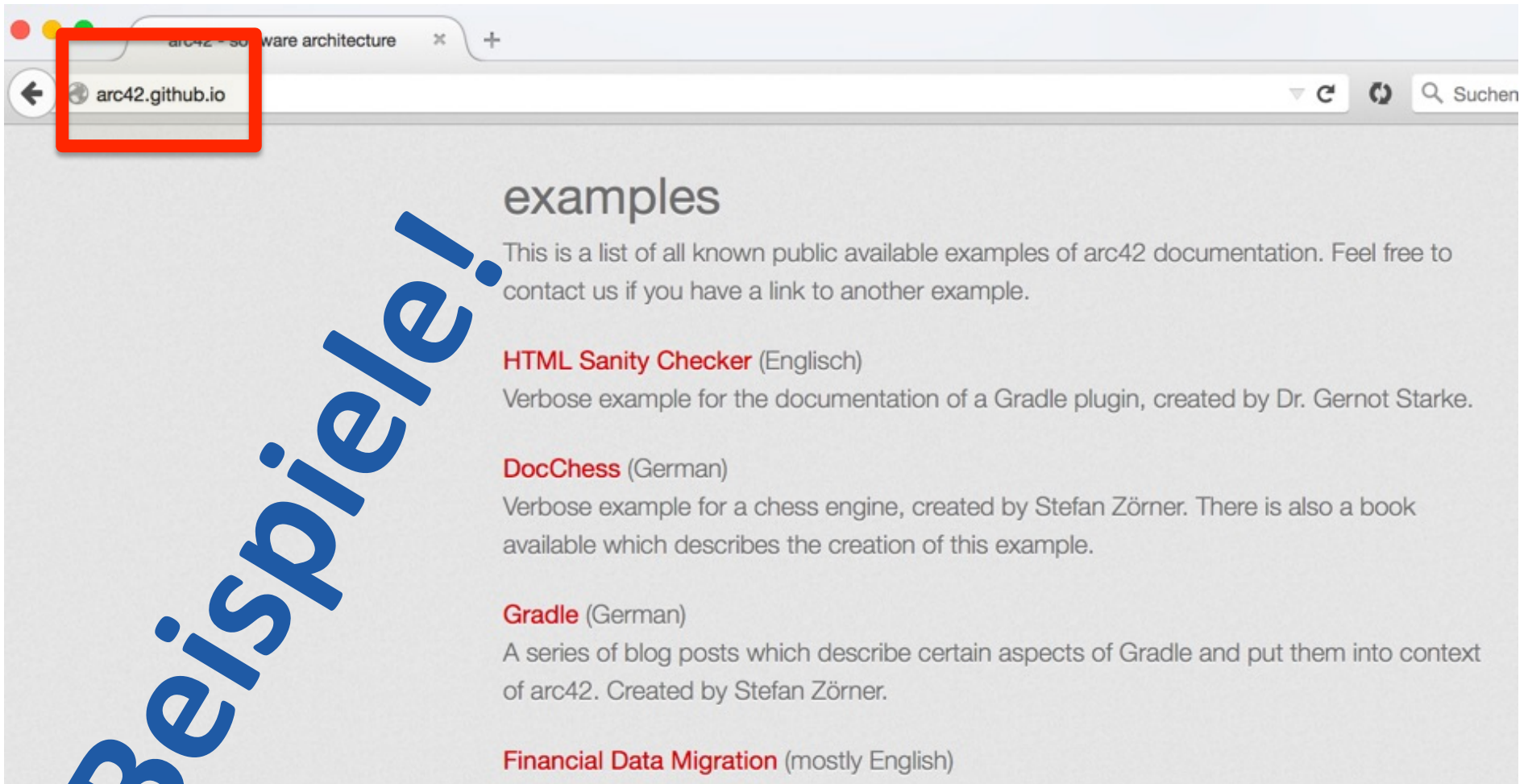
<http://aim42.github.io/htmlSanityCheck/>

innoQ⁴²-Stand 



* gerne im Tausch gegen vcard

Mit arc42 starten (Tag 1)

A screenshot of a web browser window. The address bar shows "arc42.github.io" and is highlighted with a red rectangle. The page content includes a heading "examples" and a list of documentation examples with their descriptions. A large blue watermark "Beispiele!" is overlaid on the left side of the screenshot.

arc42 - software architecture

arc42.github.io

examples

This is a list of all known public available examples of arc42 documentation. Feel free to contact us if you have a link to another example.

- HTML Sanity Checker** (Englisch)
Verbose example for the documentation of a Gradle plugin, created by Dr. Gernot Starke.
- DocChess** (German)
Verbose example for a chess engine, created by Stefan Zörner. There is also a book available which describes the creation of this example.
- Gradle** (German)
A series of blog posts which describe certain aspects of Gradle and put them into context of arc42. Created by Stefan Zörner.
- Financial Data Migration** (mostly English)

Beispiele!



Mit arc42 starten (Tag 1)



Willkommen **DokChess** Das Buch Selber starten? Informationen Unterstützung? Kontakt



arc42 Quelltexte Test-Suite javadoc

- > 1. Einführung
- > 2. Randbedingungen
- > 3. Kontextabgrenzung
- > 4. Lösungsstrategie
- > 5. Bausteinsicht
- > 6. Laufzeitsicht
- 7. Verteilungssicht
- 8. Konzepte
- 9. Entscheidungen
- 10. Szenarien
- 11. Risiken
- 12. Glossar

DokChess als Beispiel für arc42

Diese Seiten beschreiben die Architektur des Schach-Programmes DokChess. Ich habe es als Anschauungsmaterial für Vorträge und Seminare rund um Softwarearchitektur und -entwurf konzipiert und implementiert. Es dient als Fallbeispiel in meinem [Buch](#) über Architekturdokumentation.

Architekturüberblick DokChess

Dieser Architekturüberblick lässt Sie die maßgeblichen Entwurfsentscheidungen nachvollziehen. Er zeigt die Struktur der Lösung und das Zusammenspiel zentraler Elemente. Die Gliederung der Inhalte erfolgt nach der arc42-Vorlage. Zielgruppe dieses Überblicks sind in erster Linie Softwarearchitekten, die Anregungen und Beispiele suchen, wie man Architekturentwürfe angemessen dokumentieren kann. Entwickler, die selbst ein Schachprogramm schreiben wollen, erhalten wertvolle Tipps und lernen en passant einiges über methodische Softwarearchitektur.



Idee + Ausführung: Stefan Zörner (<http://embarc.de>)

Mehr Beispiele!



<http://confluence.arc42.org:>

- CRM-System (2000+ PT)
- Datenmigration (4000+ PT)

Mit arc42 starten (Tag 1, 13:30h)...

neue Systeme

1. Kontext
2. Qualitätsziele
3. Lösungsstrategie
4. Konzepte
 1. Domäne
 2. Persistenz, UI etc.

bestehende Systeme

1. Bausteine (Code-Struktur)
Level 1
2. Konzepte + typische Lösungsmuster
3. Externe Schnittstellen
(= Kontext)

Mit arc42 im Projekt?



Mit arc42 im Projekt...

Team



Projektdokumentation

„Gärtner“



Systemdokumentation



Mit arc42 im Projekt...

Team



„Gärtner“



Projektdokumentation

~arc42 „locker“

Diskussionen

Implementation-Guide,

Tasks / Issues

Systemdokumentation

1. Einführung und Ziele
2. Randbedingungen
3. Kontextabgrenzung
4. Lösungsstrategie
5. Bausteinsicht
6. Laufzeitsicht
7. Verteilungssicht
8. Konzepte
9. Entwurfsentscheidungen
10. Qualitätsszenarien
11. Risiken
12. Glossar

Weitere relevante Infos...

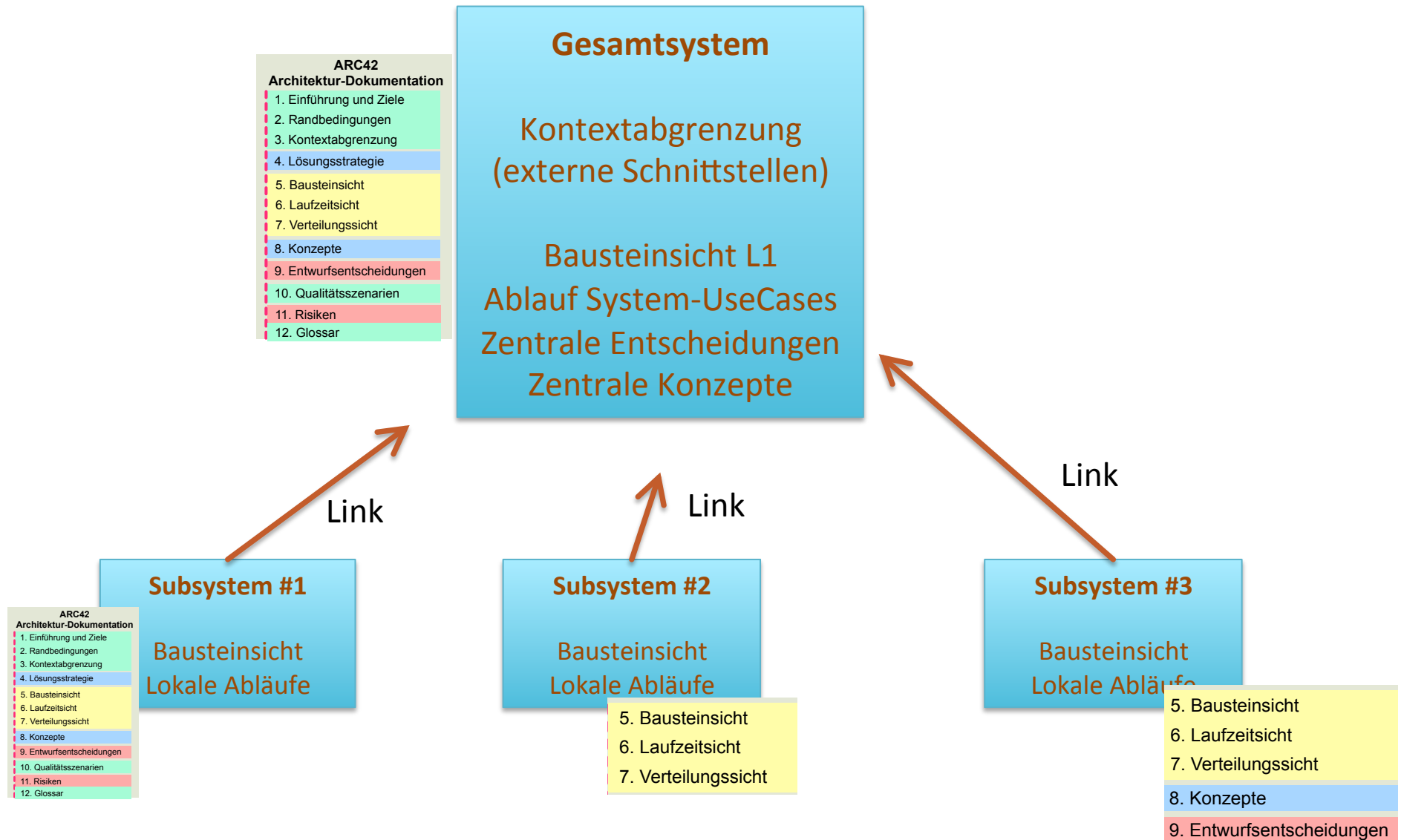
(Betrieb/Admin, Test, Release...)



Große Systeme mit arc42?



arc42 und größere Systeme



Kleine Systeme mit arc42?



Kleine Systeme...

Leeres Template == 26 Seiten
Übertrieben für kleine Projekte, oder?



Kleine Systeme...

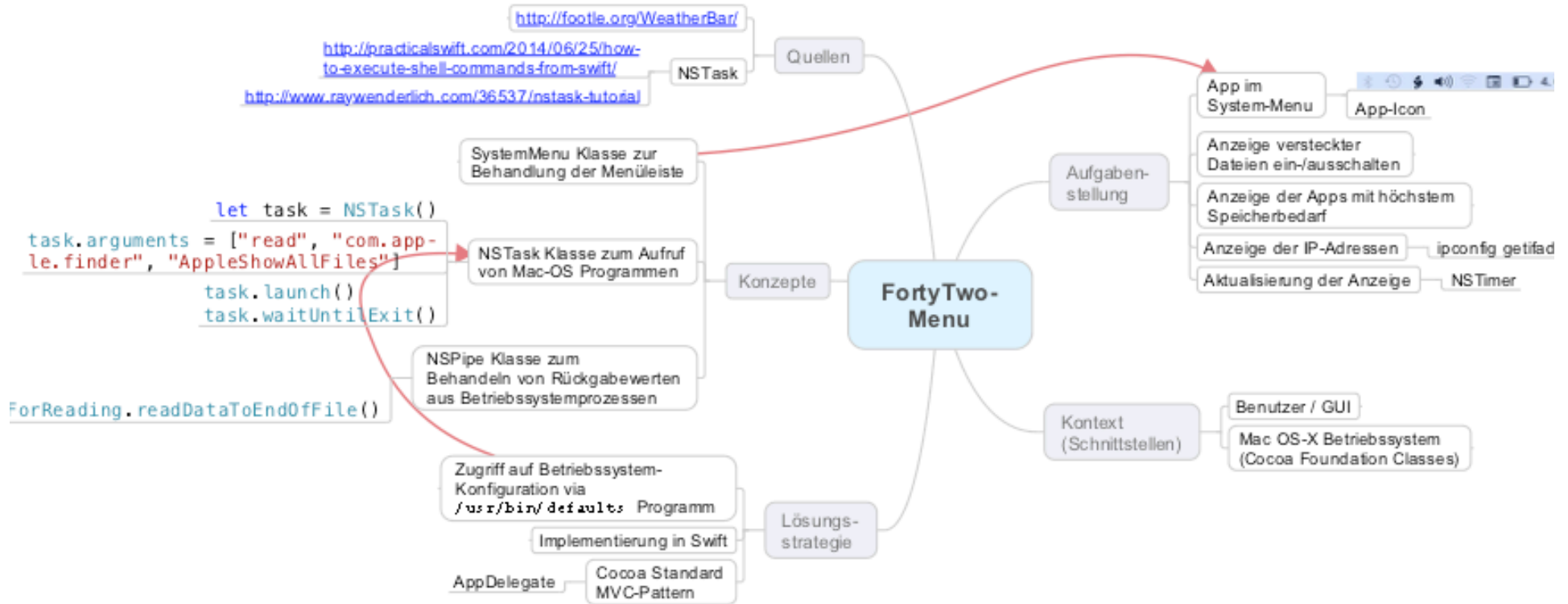
~~Leeres Template == 26 Seiten
Übertrieben für kleine Projekte, oder?~~

arc42 == **Schrank**, nicht Formular

SUPER für kleine Projekte:

- *timeboxed* verwenden
- essentielle Dinge einbringen

Beispiel „kleines System“



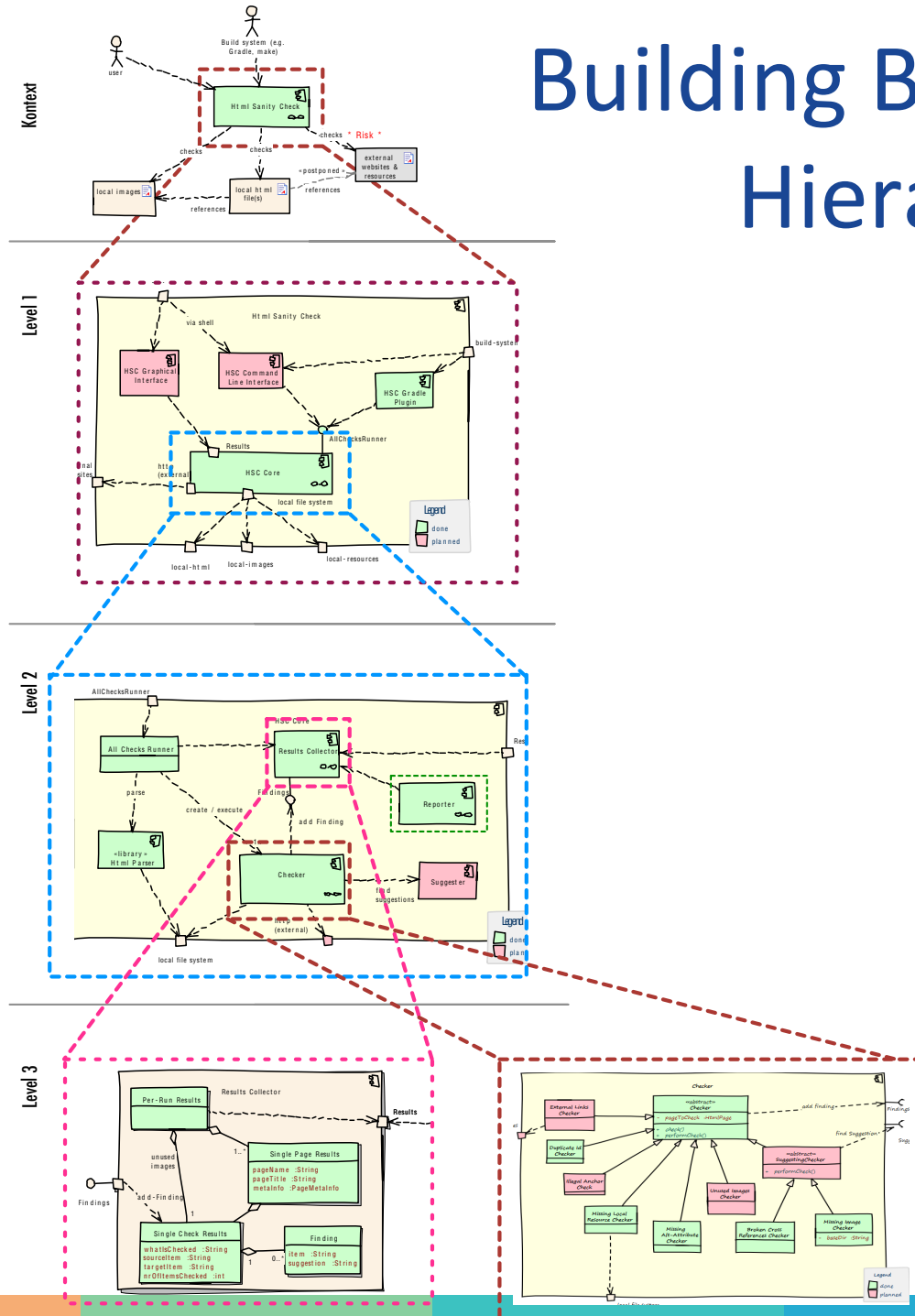
Code und Bausteinsicht?



Building Blocks Hierarchy

Gradle sub-projects:

- **core:**
org.aim42.htmlsanitycheck
- **gradle-plugin:**
org.aim42.htmlsanitycheck



Building Blocks Hierarchy

Gradle sub-projects:

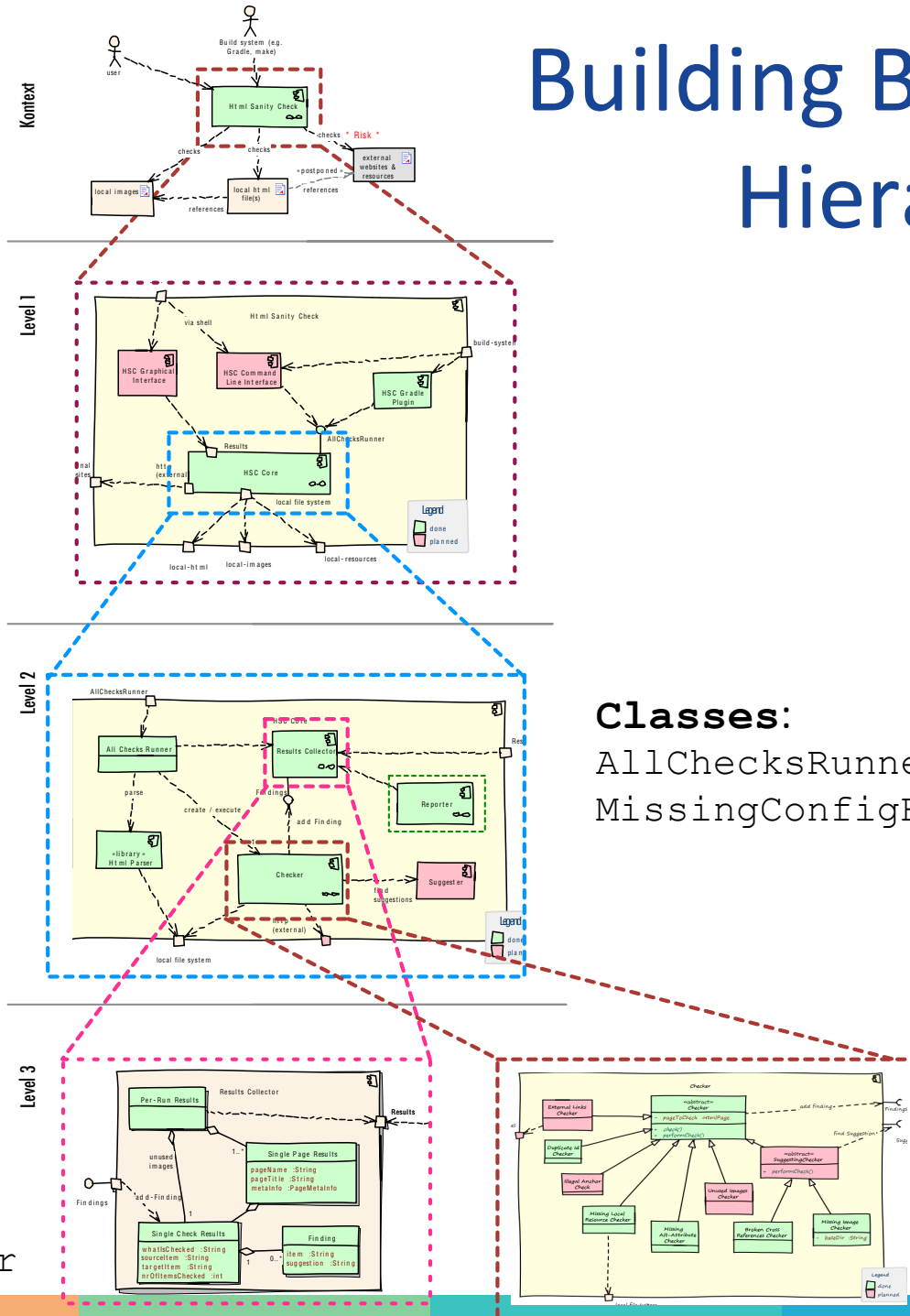
- **core:**
org.aim42.htmlsanitycheck
- **gradle-plugin:**
org.aim42.htmlsanitycheck

Packages:

- o.a.h.check
- o.a.h.collect
- o.a.h.html
- o.a.h.report

Classes:

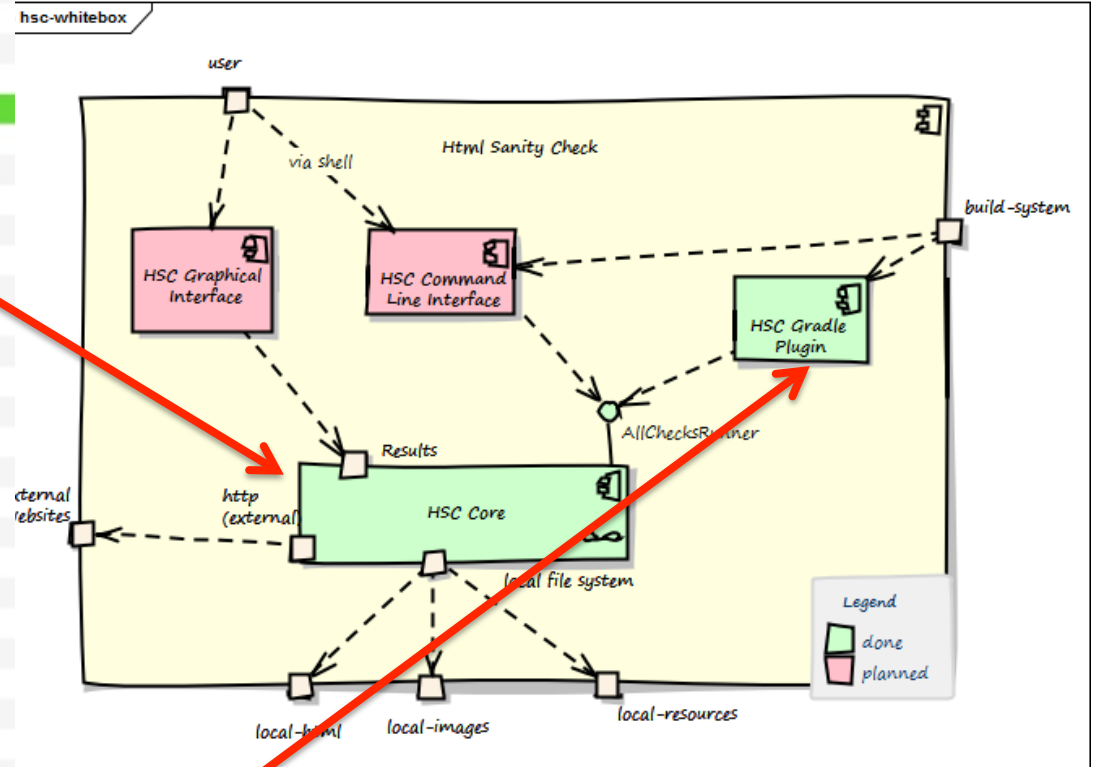
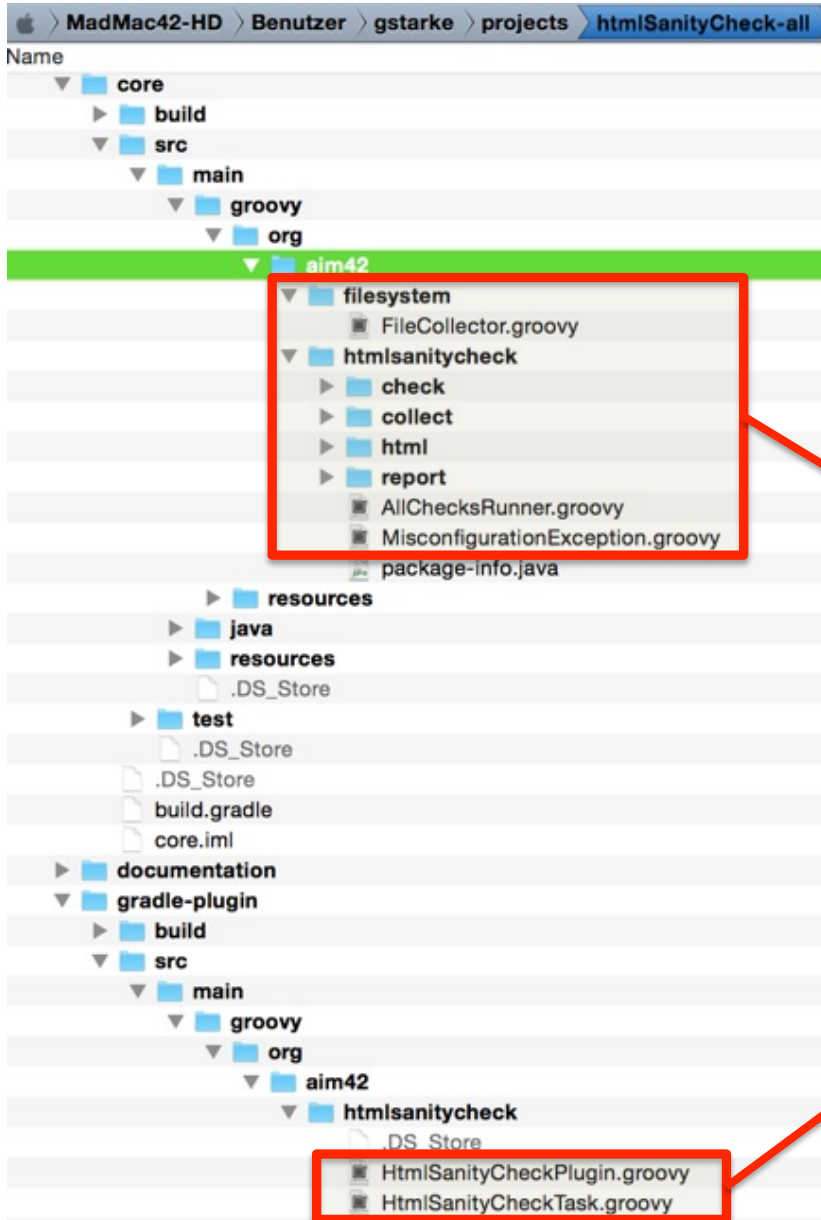
- Checker.groovy
- ImageMapChecker
- MissingImageFilesChecker



Classes:

- AllChecksRunner
- MissingConfigException

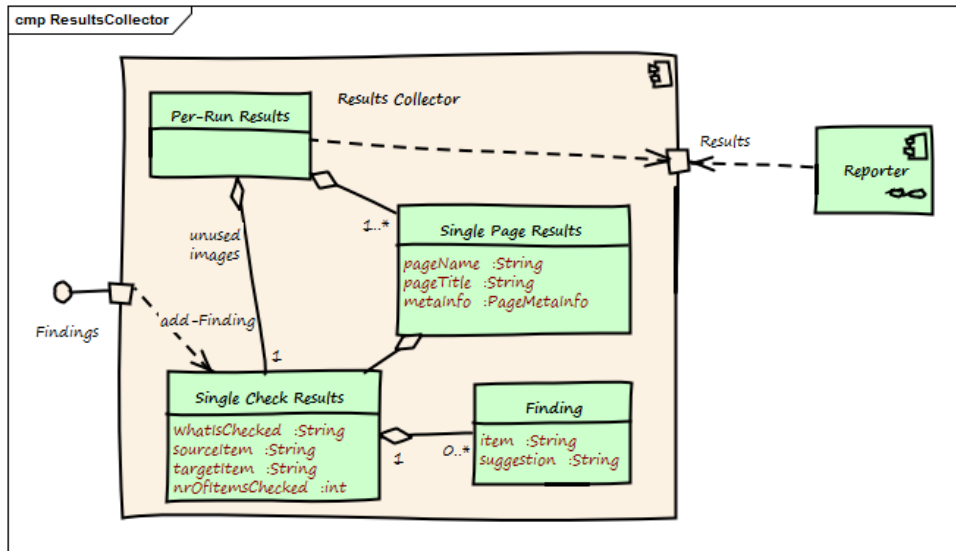
Level-1: Codestruktur im Großen



Code in der Dokumentation?



Code in Doku: Beispiel...



Wichtig: **NCP**
(never copy-paste)

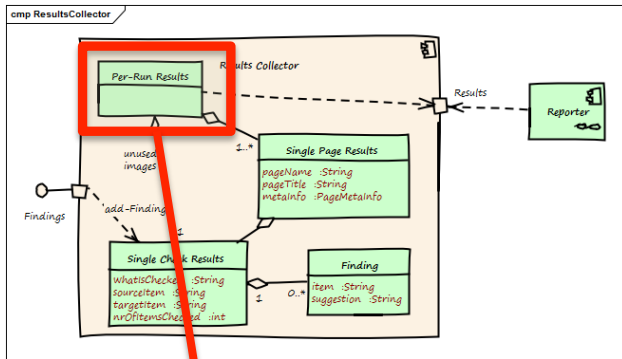
Contained Blackboxes

Table 13. ResultsCollector building blocks

Per-Run Results	results for potentially many Html pages/documents.
Single-Page-Results	results for a single page
Single-Check-Results	results for a single type of check (e.g. missing-images check)
Finding	a single finding, (e.g. "image 'logo.png' missing"). Can hold suggestions and (planned for future releases) the responsible html element.



Code in Doku: Beispiel...



Interface RunResults

```
public interface RunResults {

    // returns results for all pages which have been checked
    public ArrayList<SinglePageResults> getResultsForAllPages()

    // how many pages were checked in this run?
    public int nrOfPagesChecked()

    // how many checks were performed in all?
    public int nrOfChecksPerformedOnAllPages()

    // how many findings (errors and issues) were found in all?
    public int nrOfFindingsOnAllPages()

    // how long took checking (in milliseconds)?
    public Long checkingTookHowManyMillis()

}
```

asciidoc-Source

```
[source, groovy]
.Interface RunResults
----
include:: {coresourcepath}/htmlsanitycheck/collect/
RunResults.groovy [tags=RunResultInterface]
----
```

Konzepte!



Was ist ein Konzept?

Beispiele im Bauwesen:

- Fenster und –griffe
- Heizung / Klima
- Flach-/Walm-/Spitzdach

Was ist ein Konzept?

Plan zur Lösung von

- mehrfach auftretenden,
- übergreifenden

Problemen.

Beispiele:

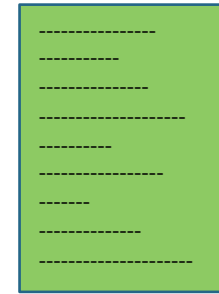
- ▶ Persistenz
- ▶ Reporting
- ▶ Batch
- ▶ Validierung
- ▶



Themen finden

Gliederung nach S. Zörner

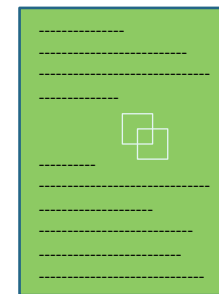
- ▶ Architekturmuster
- ▶ Entwicklung / Weiterentwicklung
 - Codegenerierung, Buildmanagement, Tests, Migration, Konfiguration, Modellierung
- ▶ Unter-der-Haube
 - Persistenz, Verteilung, Sicherheit, Transaktionen, Sessions, Caching, Parallelisierung, Threading, Geschäftsregeln, Batchverarbeitung
- ▶ Interaktion
 - UI, i18n, Ergonomie, Validierung, Barrierefreiheit, Plausibilisierung, Ausnahme-/Fehlerbehandlung, Ablaufsteuerung, Kommunikation, Integration, Reporting
- ▶ Betrieb



Identifizieren



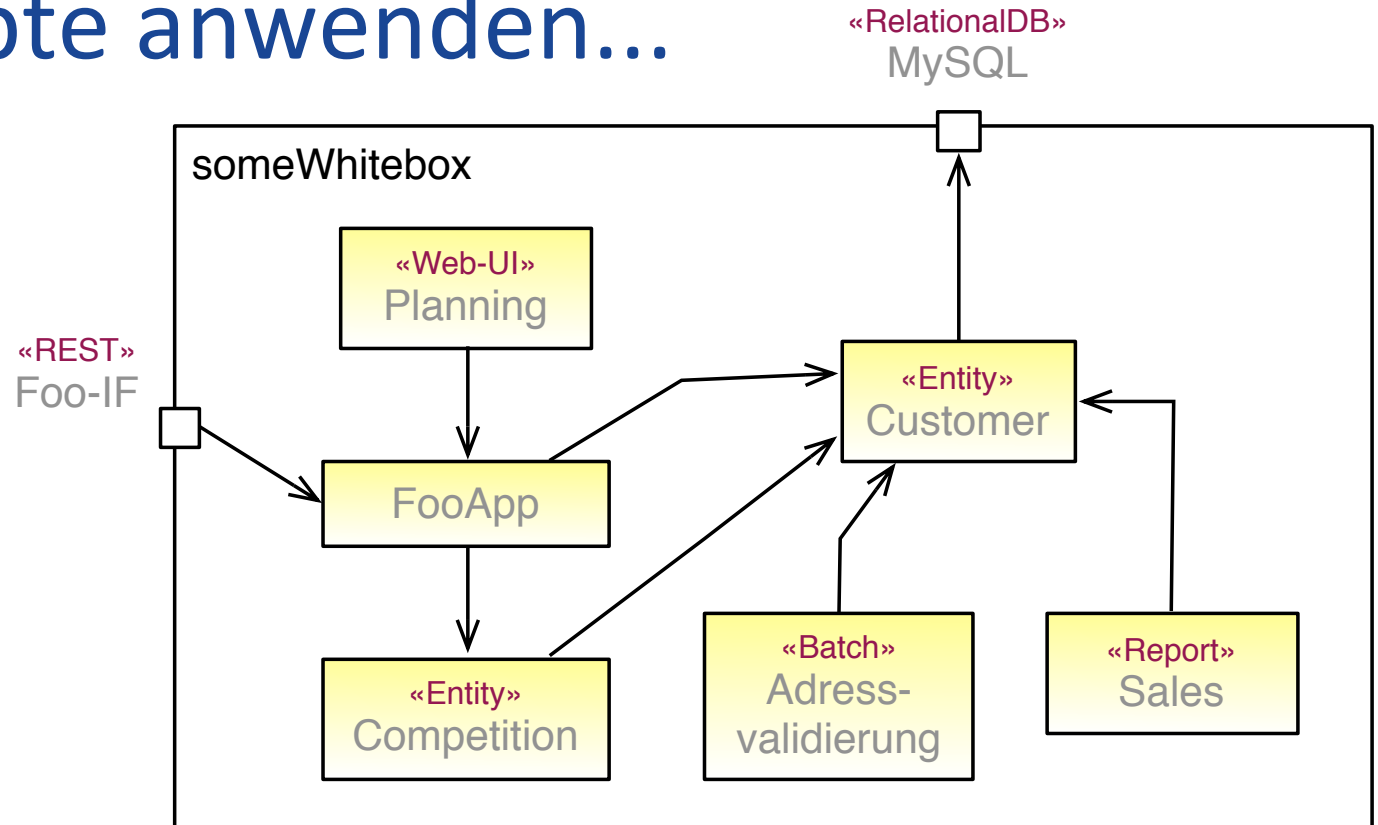
Priorisieren



ausführen



Konzepte anwenden...



Bausteinsicht:

- Zeigt Stereotypen
(=> Verweise auf Konzepte)

Konzepte:

- erklären Implementierung, mit Beispielen

Werkzeuge!



Anforderungen an Tools

- Text + Tabellen
- Diagramme
- Verweise

- Multi-User
- Änderungshistorie
- Versioniert

Variante 1: plain-Wiki

- Text + Tabellen
- Diagramme
- Verweise



- Multi-User
- Änderungshistorie
- Versioniert



Variante 1b: plain-Wiki++

- Text + Tabellen
- Diagramme
- Verweise

Zeichen-Tool (Visio,
Graffle & Co)

- Multi-User
- Änderungshistorie
- Versioniert



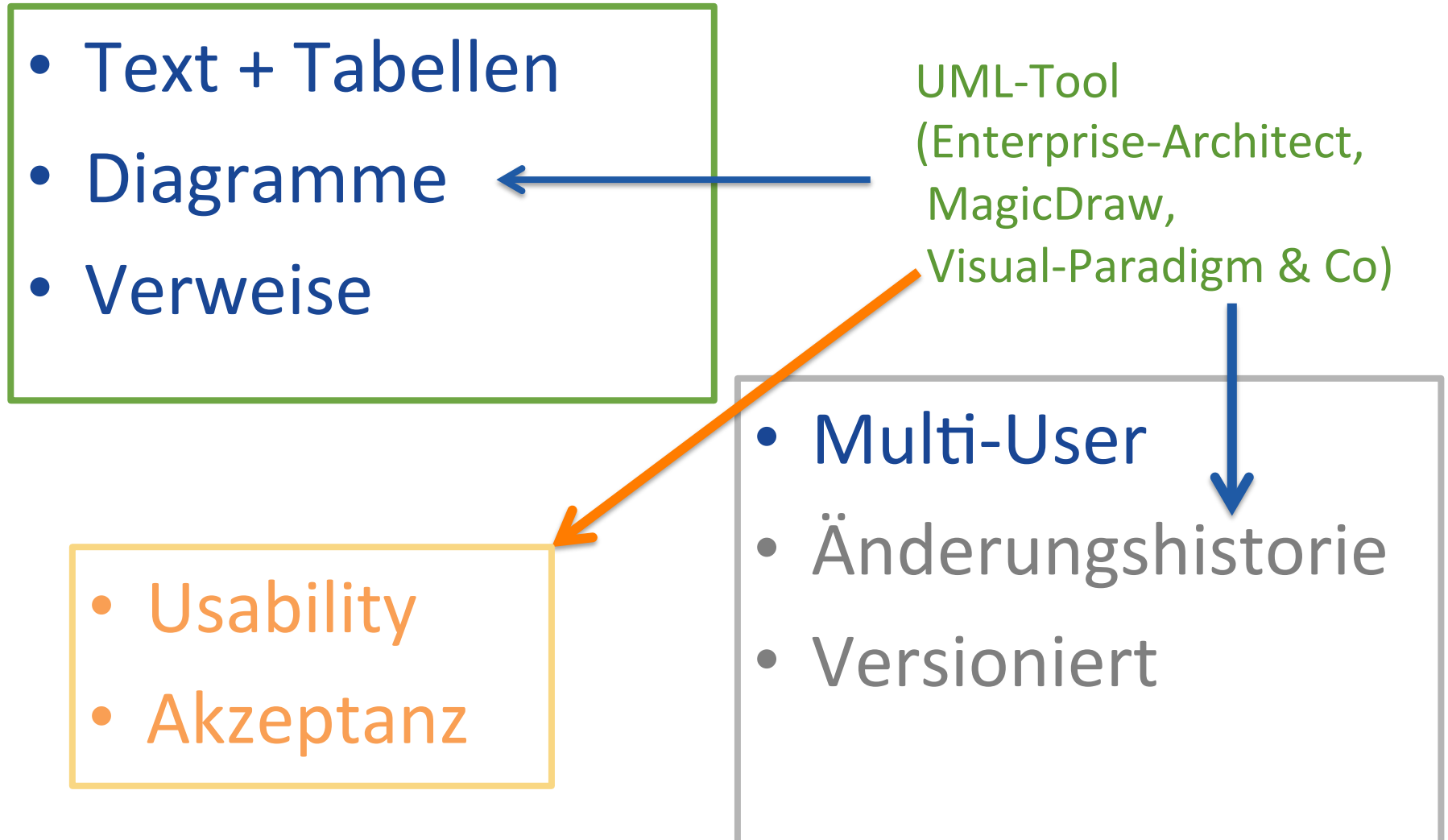
Variante 1c: Wiki + Grafik-Plugin

- Text + Tabellen
- Diagramme
- Verweise

Etwa: Gliffy
(für Confluence)

- Multi-User
- Änderungshistorie
- Versioniert

Variante 2: Wiki + UML-Tool



Variante 2: Plaintext + UML-Tool

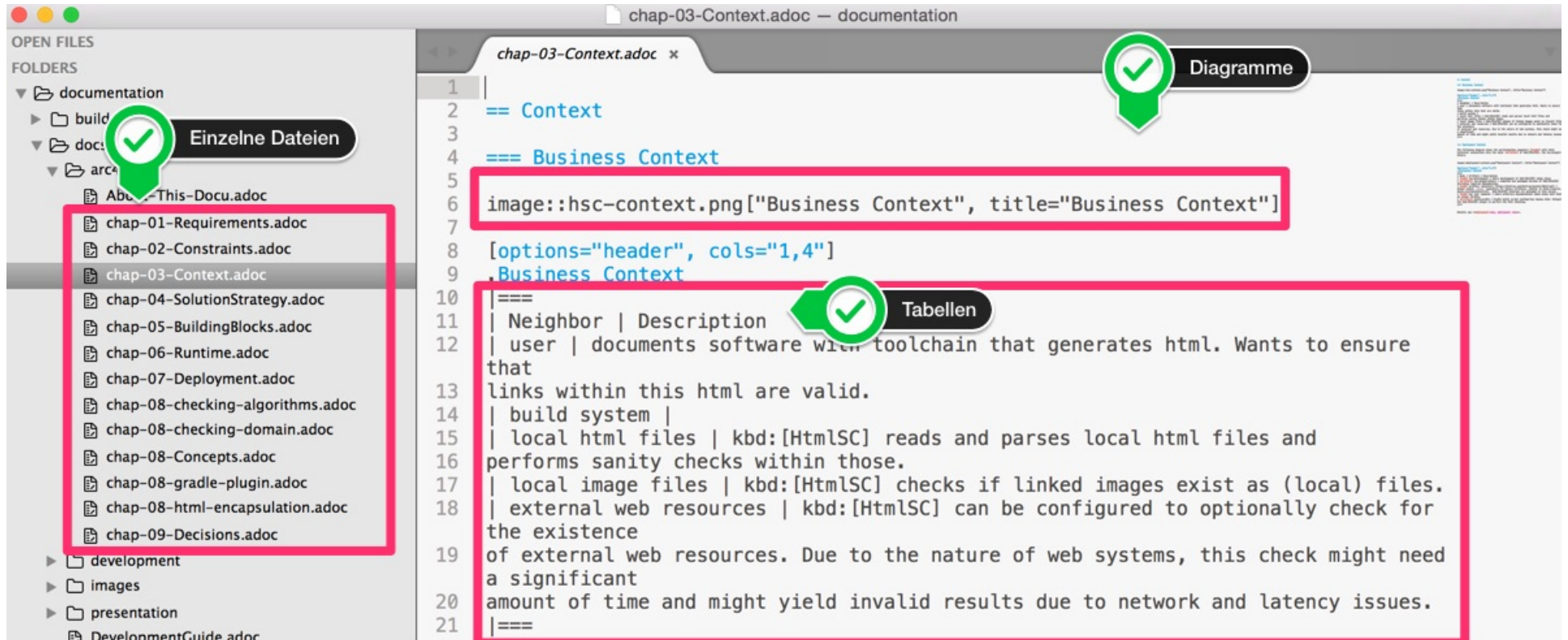
- Text + Tabellen
- Diagramme
- Verweise

- Geek-Faktor

AsciiDoc, Markdown

- Multi-User
- Änderungshistorie
- Versioniert

AsciiDoc...



The screenshot shows an IDE window titled 'chap-03-Context.adoc - documentation'. The left sidebar shows a file tree with 'documentation' as the root. Under 'documentation', there are folders 'build', 'doc', and 'arc'. The 'arc' folder contains several files, with 'chap-03-Context.adoc' selected. A red box highlights the 'arc' folder and its contents. A green checkmark icon is placed over the 'arc' folder, with a callout box labeled 'Einzelne Dateien' (Individual Files).

The main editor shows the content of 'chap-03-Context.adoc'. The code is as follows:

```

1
2 == Context
3
4 === Business Context
5
6 image::hsc-context.png["Business Context", title="Business Context"]
7
8 [options="header", cols="1,4"]
9 .Business Context
10
11 |===
12 | Neighbor | Description
13 | user | documents software with toolchain that generates html. Wants to ensure
14 | that
15 | links within this html are valid.
16 | build system |
17 | local html files | kbd:[HtmlSC] reads and parses local html files and
18 | performs sanity checks within those.
19 | local image files | kbd:[HtmlSC] checks if linked images exist as (local) files.
20 | external web resources | kbd:[HtmlSC] can be configured to optionally check for
21 | the existence
22 | of external web resources. Due to the nature of web systems, this check might need
23 | a significant
24 | amount of time and might yield invalid results due to network and latency issues.
25 |===

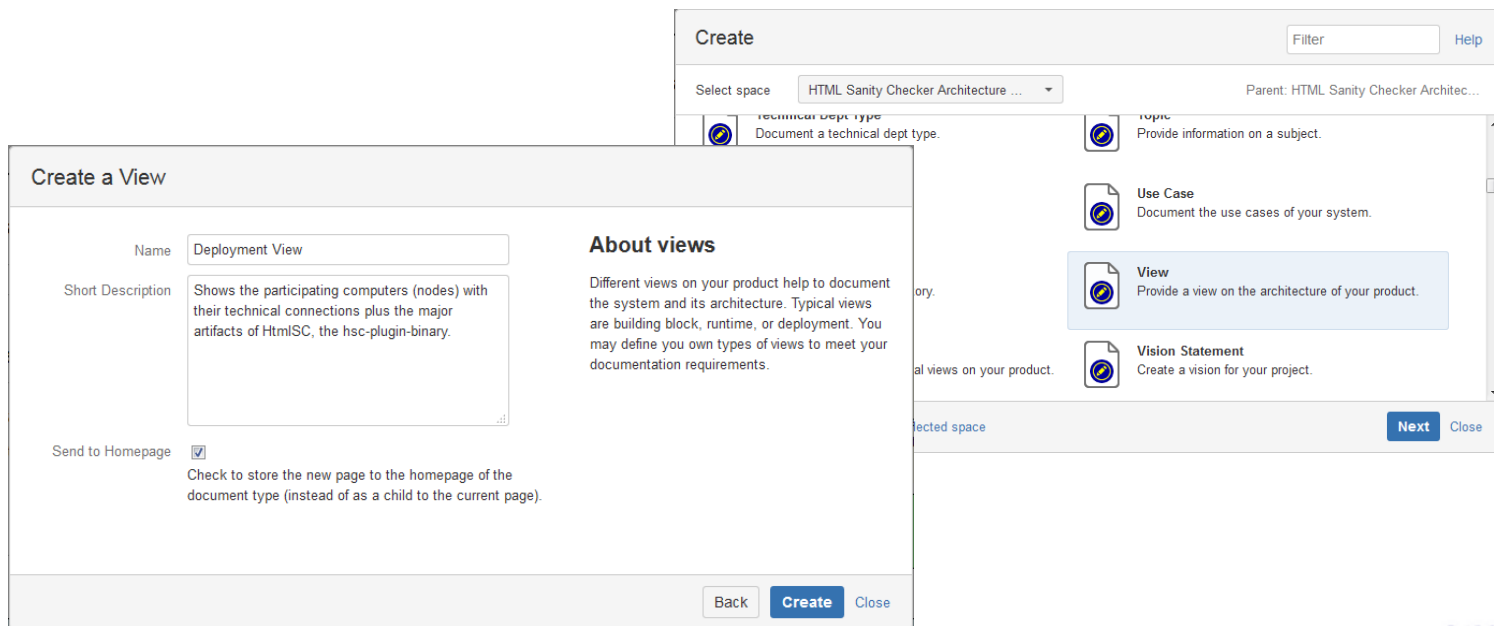
```

A red box highlights the code from line 6 to line 25. A green checkmark icon is placed over the code, with a callout box labeled 'Diagramme' (Diagrams) pointing to line 6 and 'Tabellen' (Tables) pointing to line 11.

- Plaintext (git, diff)
- Build z.B. mit Gradle
- Zielformate: html, epub, pdf

Confluence + ProjectDoc (kommerziell)

- Idee: flüssigeres Arbeiten dank Wizards



Create a View

Name:

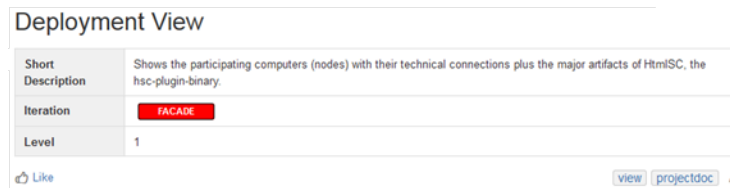
Short Description: Shows the participating computers (nodes) with their technical connections plus the major artifacts of HtmlSC, the hsc-plugin-binary.

Send to Homepage: Check to store the new page to the homepage of the document type (instead of as a child to the current page).

About views

Different views on your product help to document the system and its architecture. Typical views are building block, runtime, or deployment. You may define you own types of views to meet your documentation requirements.

Buttons: Back, Create, Close



Deployment View	
Short Description	Shows the participating computers (nodes) with their technical connections plus the major artifacts of HtmlSC, the hsc-plugin-binary.
Iteration	FACADE
Level	1

Like view projectdoc



arc42 mit projectdoc-Blueprints

- Vorlagen als Schreibhilfen

Deployment View

Document Properties Marker

Short Description	Shows the participating computers (nodes) with their technical connections plus the major artifacts of HtmlSC, the hsc-plugin-binary.	
Doctype	view	hide
Name	Deployment View	
Parent	Parent Property Name	
Audience	Name List role Audience	
Categories	Name List category Categories	
Tags	Tag List Tags	
Iteration	FACADE	
Type	Name List view-type Type	
Level	Level Macro	
Sort Key	Add a character sequence to support sorting documents.	hide

Context

Provide context information to understand the view documentation.

Unrestricted

Preview Save Close

Deployment View

Short Description	Shows the participating computers (nodes) with their technical connections plus the major artifacts of HtmlSC, the hsc-plugin-binary.
Iteration	FACADE
Level	1

Like view projectdoc

properties

actions



HTML Sanity Checker mit projectdoc

Software Architecture Documentation

Short Description
Provides detailed information about the architecture of the system.

Overview
Overview - Software Architecture Documentation

Singlepage
SWAD (Single Page)

Goals of this Documentation

This documentation is an example of arc42 documentation.
You may copy this documentation or parts of it for your own projects. In such cases you must include a link or reference to arc42 or aim42 (we regard this as fair-use).
For real-world projects, the relation of code and documentation is oversized.

Disclaimer

We provide absolutely **no guarantee**, neither for the accuracy of this documentation nor for any property or feature of the software described here.
Do not use this software in critical situations or projects.

#	Name	Short Description
1	Introduction and Goals	The introduction to the architecture documentation lists the driving forces that software architects must consider in their decisions. This includes on the one hand the fulfillment of functional requirements of the stakeholders, on the other hand the fulfillment of or compliance with required constraints, always in consideration of the architecture goals.
2	Architecture Constraints	Lists any requirement that constrains software architects in their freedom of design decisions or the development process.
3	System Scope and Context	Describes the context view that defines the boundaries of the system under development to distinguish it from neighboring systems.
4	Solution Strategy	Provides a short summary and explanation of the fundamental solution ideas and strategies.
5	Building Block View	Provides a static decomposition of the system into building blocks and the relationships between them.
6	Runtime View	Describes the behavior and interaction of the systems building blocks as runtime elements.
7	Deployment View	Describes the environment within which the system is executed.
8	Concepts	Covers examples of frequent cross-cutting concerns.
9	Design Decisions	Documents all important design decisions and their reasons.
10	Quality Scenarios	Summarizes all relevant scenarios to systematically evaluate the architecture against the quality requirements.
11	Technical Risks	Lists the identified technical risks, ordered by priority.
12	Glossary	Lists the most important terms of the software architecture in alphabetic order.

Like Be the first to like this swad projectdoc

HTML Sanity Checker Architecture Documentation

Search this space

Featured Pages

- Business Context
- Deployment Context
- hsc-plugin-binary
- SWAD (Single Page)

Documentation

- Tours - compiled topics.
- FAQs - questions and answers.
- Topics - information on project topics.
- Glossary - terms used or defined.

Project Library

- Resources - external media.
- Quotes - relevant for the project.

Team & Stakeholders

- Stakeholders - working on the project.
- Persons - related to the project.
- Organizations - related to the project.

References

- Roles - of all stakeholders.
- Categories - browse docs by category.
- Tags - browse docs by tag.
- Types - defined for doctypes.

Agile Documentation

- Modules - content for single-sourcing.
- Documentation Dashboard - team collaboration tool.

More on documentation?

- arc42 - software architecture documentation
- arc42 on GitHub - asciidoc & more
- projectdoc - add-on for Confluence

Volumes

- Vision Statement
- Architecture Vision
- Software Architecture Overview
- Software Architecture Documentation
- Performance Management
- Security Management
- Test Management
- Usability Management

Analysis

- Vision Statements - of project or iterations.
- Features - implemented by the products.
- Out Items - not covered by this project.
- User Characters - performing in stories.
- Project Constraints - that cannot be changed.
- Requirements - to be implemented.
- Quality Targets - relevant for the project.
- Quality Scenarios - to evaluate the system.

Design

- Components - part of the system.
- Views - on the system.
- Use Cases - of the product.
- Architecture Aspects - of the system.
- Architecture Decisions - made.
- Technical Depts - to be tracked.

Implementation

- API Documentation
- Product Codes
- Properties - of the product.
- Reports - static analysis et al.
- Releases / Build Metadata
- Changes

Test

- Unit Tests
- Integration Tests
- Performance Tests
- Security Tests
- Acceptance Tests
- Usability Tests

Deployment

- Artifacts - created by the build process.
- Environments - running artifacts.
- Nodes - deployment targets.

Like

projectdoc



Variante 3: UML-Tool

- Text + Tabellen
- **Diagramme**
- Verweise

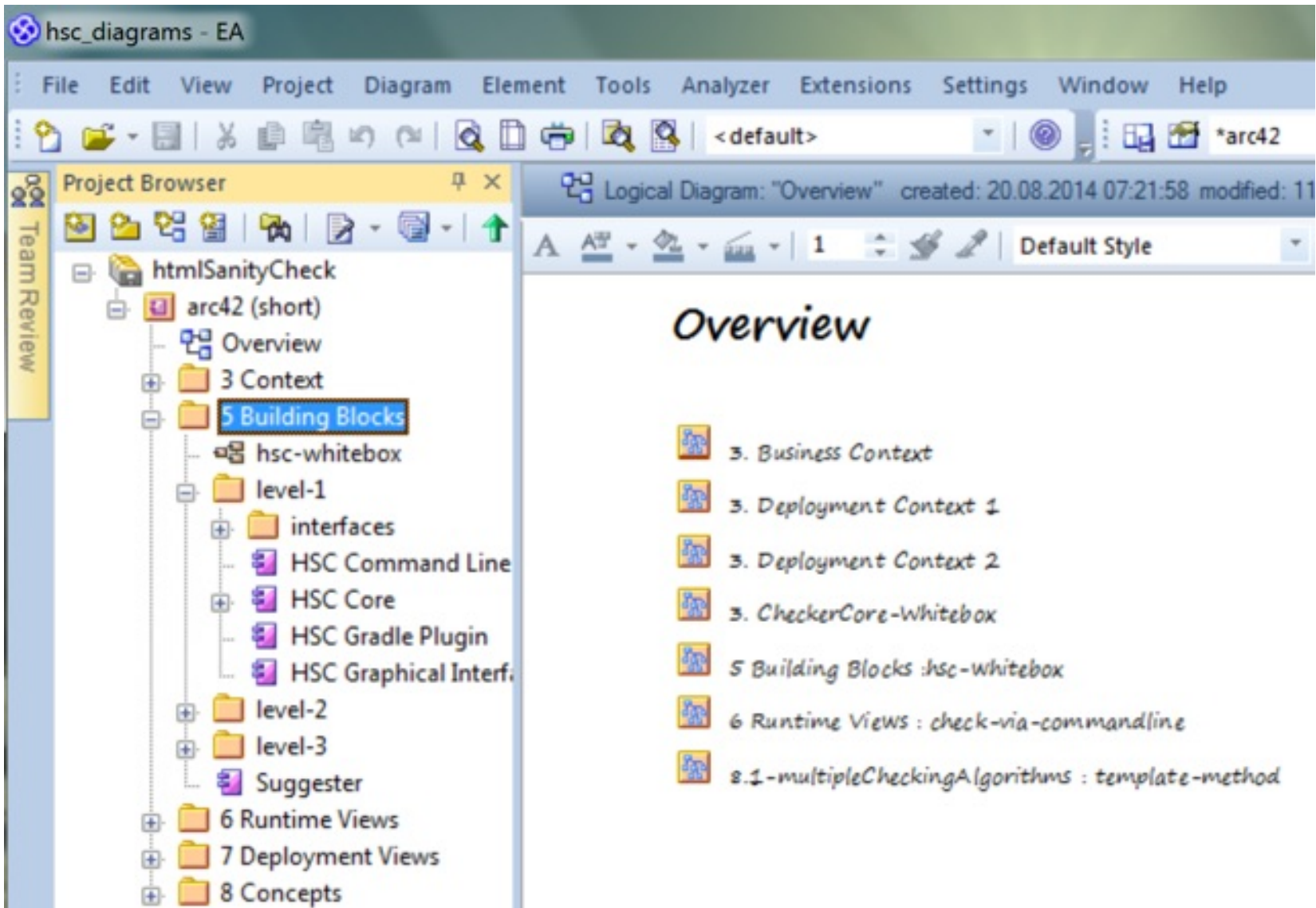
← Modelle

- **Geek-Faktor**

Enterprise-Architect,
Visual Paradigm etc.

- Multi-User
- **Änderungshistorie**
- **Versioniert**

Beispiel: arc42 mit EA



The screenshot shows the 'hsc_diagrams - EA' application interface. The top menu bar includes File, Edit, View, Project, Diagram, Element, Tools, Analyzer, Extensions, Settings, Window, and Help. The toolbar contains various icons for file operations and diagram manipulation. The Project Browser on the left shows a tree structure for the 'htmlSanityCheck' project, with 'arc42 (short)' expanded to show 'Overview', '3 Context', '5 Building Blocks', 'hsc-whitebox', 'level-1', 'level-2', 'level-3', and 'Suggester'. The '5 Building Blocks' folder is highlighted. The main diagram area displays a 'Logical Diagram: "Overview"' with a list of elements:

- 3. Business Context
- 3. Deployment Context 1
- 3. Deployment Context 2
- 3. CheckerCore-Whitebox
- 5 Building Blocks :hsc-whitebox
- 6 Runtime Views : check-via-commandline
- 8.1-multipleCheckingAlgorithms : template-method

UML-Tool + Export-Automatisierung

- Export aus Enterprise-Architect™
 - Skript in github.com/arc42 vorhanden

- Export aus VisualParadigm™
 - Export möglich
 - bislang kein OpenSource Skript verfügbar



Wohin damit?



Wohin mit Domain-Model?

1. (gut) Querschnittliche Konzepte
2. (ok) Bausteinsicht („Domain-Component“)
3. (hhm) Eigenes (neues) Kapitel



Wohin mit UI/Layout, Masken oder Screenflow?

1. Querschnittliche Konzepte
2. Eigene Sicht
 - (== neues Kapitel)





Kata mit arc42

(„Hands-on Workshop“)



Ziele einer Architektur-Kata

- Gemeinsames Verständnis
- Aufgaben „erproben“
- Aspekte von „Architektur“ erarbeiten



Kata mit arc42

Aufgabe

Anforderungen
Qualitätsziele

Kontext

Lösungsidee

Bausteine

Konzepte



Subprojekte!

Qualitätsanforderungen

wesentliche
Architekturaufgaben



Beispiele für Q-Anforderungen

- Q-Szenarien
 - Etwa 60 praxisnahe Qualitätsziele
(aus konkreten Projekten/Systemen abstrahiert)
- Definitionen für typische Q-Anforderungen
 - 70+ häufig vorkommende Begriffe rund um Qualität – pragmatisch definiert



Beispiele für Q-Anforderungen

Beispiele für Qualitätsanforderungen an Software

Dr. Gernot Starke – gernot.starke@innoq.com – v0.6

Über Qualität

Die Qualität eines Produktes oder Systems ist ganz allgemein als Menge von Eigenschaften oder Merkmalen definiert.

In der Praxis haben sich einige Kategorien (Oberbegriffe) für häufig auftretende *Qualitätsanforderungen* (synonym: Qualitätsziele) etabliert, im wesentlichen durch das DIN/ISO 9126 Begriffsmodell geprägt.

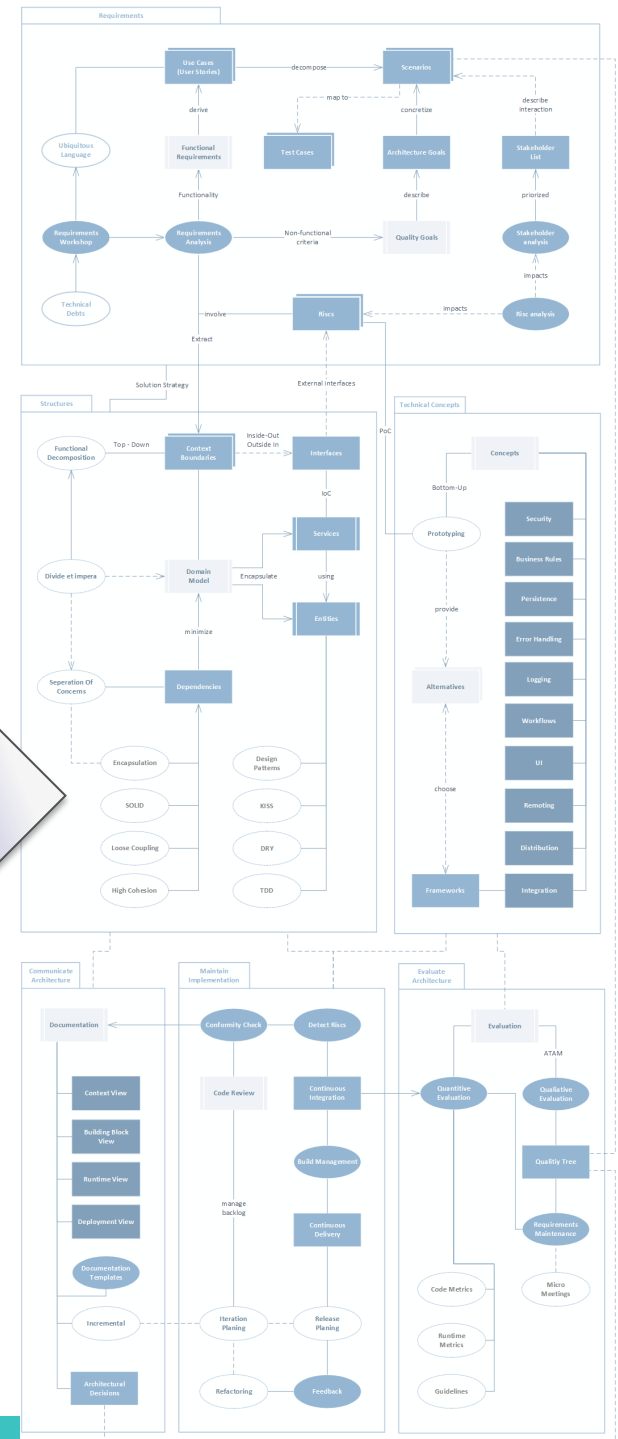
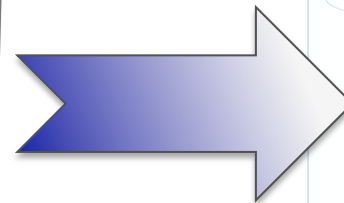
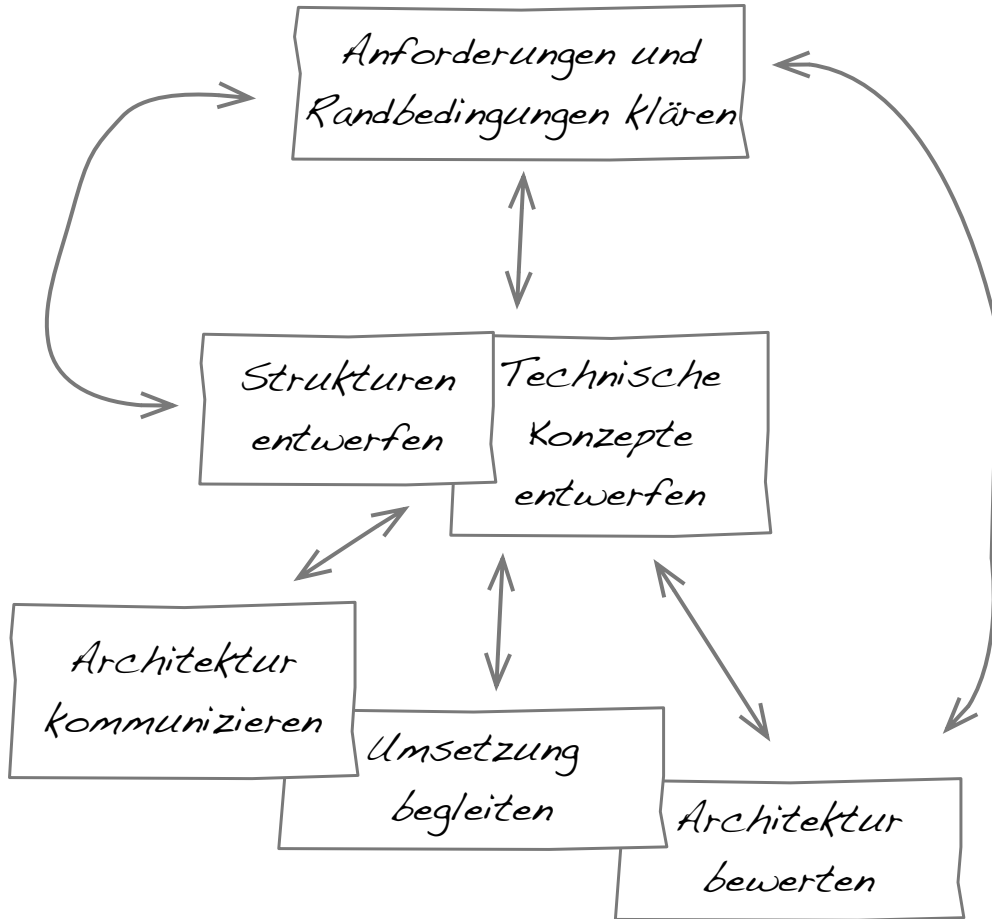
Die Kategorien

- [Änderbarkeit](#)
- [Benutzbarkeit](#)
- [Effizienz](#)
- [Zuverlässigkeit](#)
- [Betreibbarkeit](#)
- [Sonstiges \(u.a. Funktionalität\)](#)

Demo...



Architektur- Aufgaben...





Dr. Gernot Starke

Gernot.Starke@innoQ.com

<http://arc42.de>
<http://innoq.com>